

Do Not Perform Arithmetic with Unvalidated Input

William L. Fithen, Software Engineering Institute [vita³]

Copyright © 2005 Carnegie Mellon University

2005-10-03

L4 / D/P⁴

Careless modulo arithmetic can introduce vulnerability.

Description

According to [Seacord 05]:

Integers represent a growing and underestimated source of vulnerabilities in C and C++ programs. This is primarily because boundary conditions for integers, unlike other boundary conditions in software engineering, have been intentionally ignored. Most programmers emerging from colleges and universities understand that integers have fixed limits, but because these limits were either deemed sufficient, or because testing the results of each arithmetic operation was considered prohibitively expensive, violating integer boundary conditions has gone almost entirely unchecked in commercial software.

For an indepth coverage of this issue in C and C++, see Safe Integer Operations⁷.

References

- [Blexim 02] blexim. *Basic Integer Overflows*. <http://www.phrack.org/phrack/60/p60-0x0a.txt> (2002).
- [Hoglund 04] Hoglund, Greg & McGraw, Gary. *Exploiting Software: How to Break Code*. Boston, MA: Addison-Wesley, 2004.
- [Horovitz 02] Horovitz, Oded. *Big Loop Integer Protection*. <http://www.phrack.org/phrack/60/p60-0x09.txt> (2002).
- [Howard 03a] Howard, Michael. *Reviewing Code for Integer Manipulation Vulnerabilities*. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure04102003.asp> (2003).
- [Seacord 05] Seacord, Robert C. *Secure Coding in C and C++*. Boston, MA: Addison-Wesley, 2005.
- [Thompson 05] Thompson, Herbert & Chase, Scott. *The Software Vulnerability Guide*. Charles River Media, 211-222. 2005.

Carnegie Mellon Copyright

Copyright © Carnegie Mellon University 2005-2010.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu¹.

3. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/320-BSI.html (Fithen, William L.)

7. <http://buildsecurityin.us-cert.gov/bsi/articles/knowledge/coding/312-BSI.html> (Safe Integer Operations)

1. <mailto:permission@sei.cmu.edu>

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

NO WARRANTY

THIS MATERIAL OF CARNEGIE MELLON UNIVERSITY AND ITS SOFTWARE ENGINEERING INSTITUTE IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.